REMARKS

Claim 27 has been amended to correct its dependency. Claims 1-28 remain pending in the present application. No new matter has been added. In view of the above amendments and the following remarks, it is respectfully submitted that all of the presently pending claims are allowable.

I. The Rejection Under 35 U.S.C. 103 Should Be Withdrawn

The Examiner rejected claims 1-28 under 35 U.S.C. 103 as being unpatentable over U.S. Patent 5,966,702 to Fresko et al. ("the Fresko reference") in view of U.S. Patent 6,339,841 to Merrick et al. ("the Merrick reference"). *Office Action*, page 2.

The Fresko reference describes a method for pre-processing and packaging class files. Specifically, Fresko describes pre-processing a set of class files into a single file called a multi-class file. *Fresko Reference,* col. 9, lines 10-14. The method is described as taking the set of classes and parsing each of the individual classes to determine the shared set of parameters shared by these classes, *e.g.*, string and numeric constants. *Id.* at col. 9, lines 15-21. These shared values are then stored in a common pool and removed from the pools of the individual classes, thereby reducing the storage requirements for the classes. *Id.* at col. 9, lines 21-25. The multi-class file is then created containing the common pool and all the individual classes having the reduced size. *Id.* at col. 9, lines 36-43. The multi-class file may then be loaded onto the machine using the virtual machine. *Id* at col. 10, lines 16-24. This multi-class loading is described as consolidating the loading of the individual files into a single transaction. *Id.* at col. 10, lines 36-51.

The Examiner states that the Fresko reference discloses "a stack to record the request" and "when the loading of the component is unsuccessful, contents of the stack are made available to a user to indicate the unsuccessfully loaded component." *Office Action*, page 2. The applicant respectfully disagrees with this characterization of the Fresko reference. The stacks described in the Fresko reference are in the runtime data areas of the virtual machine. *Fresko reference*, col. 10, lines 52-54. Specifically the Fresko reference states that "[t]he stacks are used by the respective threads to store local variables, partial results and an operand stack." *Id.* at col. 10, lines 62-64. Each of these functions relate to the runtime operation of the running application and not to the loading of classes. Specifically, the partial results referred to by the Examiner relates to the storage of partial results during runtime operation of the application.

In contrast, the present invention relates to identifying errors in the loading of components, *e.g.*, classes. The stack is used during the loading of the components to record a reference to each of the components that are to be loaded and remove the reference when the component loads correctly. Claim 1 of the present invention recites "a stack to record the request," the request being "a request to load a component." As described above, the Fresko reference loads all the classes at once while loading the multi-class file. Thus, there is no reason to record references to the classes because the Fresko reference eliminates the individual loading of each class and loads all classes via the multi-class file. There is no teaching in the Fresko reference that the disclosed stack "record[s] the request" to load a component as recited in claim 1. Accordingly, the applicant respectfully submits that the Fresko reference does not teach the claim limitation "a stack to record the request" as recited in claim 1, and therefore the rejection should be withdrawn.

Furthermore, as the Examiner correctly points out, the Fresko reference also does not teach "wherein when the request has been fulfilled the request is removed from the stack." *Office Action*, page 2. However, the Examiner states that the Merrick reference cures this failing in the Fresko reference. *Id.* at p. 2. The Merrick reference discloses a system and method for loading classes. Specifically, the Merrick reference describes that a class is comprised of various parts such as a constant pool and methods. *Merrick reference*, col. 3, line 60 - col. 4, line 7. The Merrick reference teaches that instead of loading all the byte code for a class, some byte code for the methods may be loaded when needed. *Id.* at col. 4, lines 2-4. This is done by loading a method table which includes pointers to the non-loaded methods allowing the non-loaded methods to be loaded when needed. *Id.* at col. 4, lines 39-46.

Thus, while the method table of the Merrick reference has a reference to non-loaded methods, the method differs significantly from the stack of the present invention. In the first instance the stack of the present invention "record[s] the request" to load a component. The method table of the Merrick reference does the exact opposite, it records a list of methods which have been specifically requested <u>not</u> to be loaded. In addition, there is no teaching in the Merrick reference that when the method is loaded that the reference to the method is removed from the method table.

The Examiner points to the language in the Merrick reference which states "[o]ne extreme way to discard unused components would be to discard all the methods not active on the Java stack, if the individual components were needed later they could be reloaded with ease."

Merrick reference, col. 3, lines 6-8. However, this reference to the Java stack is once again in direct conflict with the operation of the stack of the present invention. The Java stack has a record of all the methods which have been loaded. Thus, the statement means that any methods which are loaded and inactive can be discarded and then reloaded from the method table if needed at a later time. Therefore, the reference to the method is not removed from the stack when the method is loaded. In fact, the normal operation of the method call stacks created by the Java Virtual Machine and referred to by the Merrick and Fresko references, create the method reference on the stack when the method is loaded.

Accordingly, for the reasons described above the applicant respectfully submits that neither the Fresko reference nor the Merrick reference, either alone or in combination, teach, disclose or suggest "*a stack to record the request*" and/or "*when the request has been fulfilled the request is removed from the stack*" as recited in claim 1. Thus, the rejection of claim 1 and all claims depending therefrom (claims 2-7) should be withdrawn.

The Examiner also rejected claim 12 for the same reasons as described above for claim 1. *Office Action*, page 2. Claim 12 includes similar limitations as claim 1. Specifically, claim 12 recites "a stack to record a load request for a software component" and a loader which "pops the representation of the software component off of the stack when the load request has been successfully fulfilled." As described in the specification, "removing a representation of a class from a stack is referred to as 'popping' a class off the stack." *Specification*, paragraph [0014]. Accordingly, for the reasons described above with reference to claim 1, the applicant respectfully submits that the rejection of claim 12 and all claims depending therefrom (claims 13-16) should be withdrawn.

The Examiner also rejected claim 8 over the Fresko reference in view of the Merrick reference. *Office Action*, page 3-4. Claim 8 includes limitations similar to those recited in claim 1. Specifically, claim 8 recites "placing a representation of the first software module onto a stack" and "removing the representation of the first software module from the stack when the first software module has been successfully loaded." Accordingly, for the reasons described above with reference to claim 1, the applicant respectfully submits that the rejection of claim 8 and all claims depending therefrom (claims 9-11) should be withdrawn.

The Examiner also rejected claim 17 over the Fresko reference in view of the Merrick reference. *Office Action*, page 3-4. Claim 17 includes limitations similar to those recited in claim 1. Specifically, claim 17 recites "placing a representation of the first Java class onto a stack" and "removing the representation of the first Java class from the stack when the first Java class has been successfully loaded." Accordingly, for the reasons described above with reference to claim 1, the applicant respectfully submits that the rejection of claim 17 and all claims depending therefrom (claims 18-22) should be withdrawn.

The Examiner also rejected claim 23 over the Fresko reference in view of the Merrick reference. *Office Action*, page 5-6. Claim 23 includes limitations similar to those recited in claim 1. Specifically, claim 23 recites "a stack to record a load request for a Java class" and a Java class loader which "pops the representation of the Java class off of the stack when the load request has been successfully fulfilled." Accordingly, for the reasons described above with reference to claim 1, the applicant respectfully submits that the rejection of claim 23 and all claims depending therefrom (claims 24-28) should be withdrawn.

CONCLUSION

In view of the amendments and remarks submitted above, the applicant respectfully submits that the present case is in condition for allowance. All issues raised by the Examiner have been addressed, and a favorable action on the merits is thus earnestly requested.

Respectfully submitted,

Dated: February 27, 2004

By: [illegible]
Michael J. Marcin (Reg. No. 48,198)

FAY KAPLUN & MARCIN, LLP
150 Broadway, Suite 702
New York, NY 10038
(212) 619-6000 (phone)
(212) 208-6819 (facsimile)